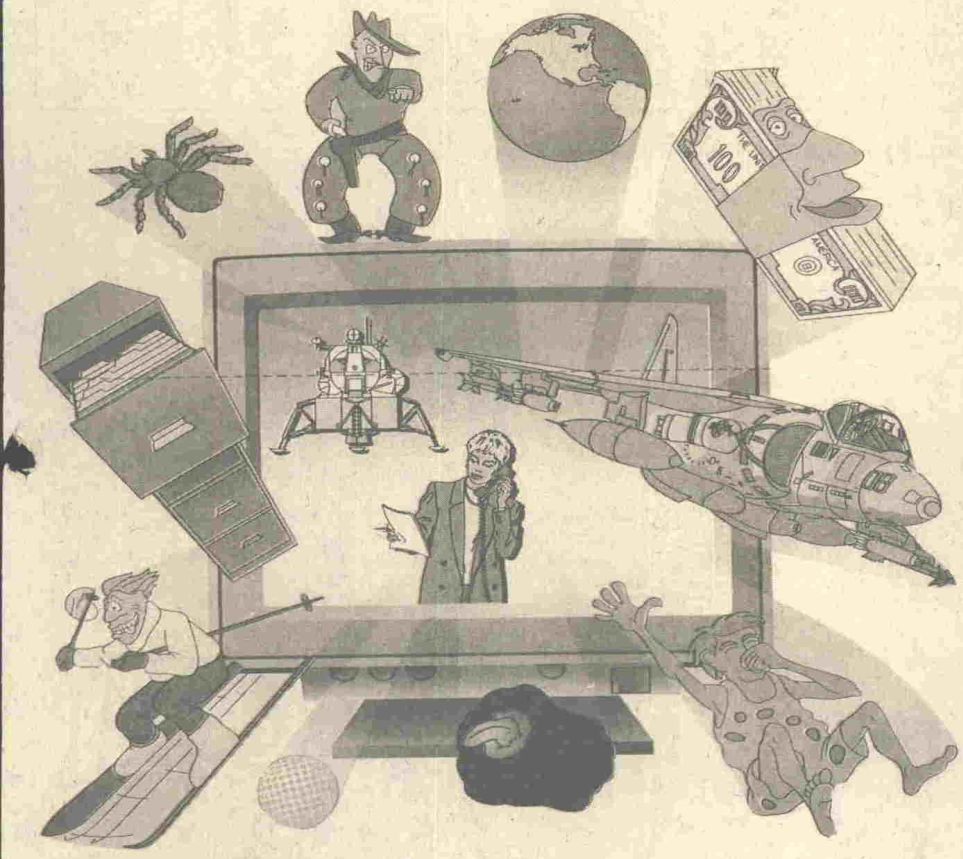
# คอมพิวเตอร์ยุค มัลติโปรเซสเซอร์



***หลายคนยังสับสนถึงคำว่า "multiprocessing" กับ "parallel processing" ต่างกันหรือเหมือนกัน ?***

มื่อเราเปรียบเทียบว่าคอมพิวเตอร์ 2 เครื่องใครจะดีกว่ากัน ความเร็วย่อมเป็นปัจจัยหนึ่งในหลายปัจจัยที่ต้องพูดถึงด้วยเสมอ และมีการพัฒนาขึ้นมาเรื่อย ๆ อย่างไมโครโปรเซสเซอร์ที่เราคุ้นเคยในเครื่องพีซีทั้งหลาย ก็เริ่มจาก 4.77 เป็น 8, 12, 25, 33 เมกะเฮิรตซ์ มีแต่ขึ้นแล้วก็ขึ้น แต่ก็ยังไม่พอต่อความเร็วที่คนเราอยากให้มันมี อย่างไรก็ตาม การใช้โปรเซสเซอร์ตัวเดียวนี้ ก็ต้องมีความเร็วที่จำกัดอยู่ที่จุด ๆ หนึ่ง ซึ่งปัจจุบันก็กำลังจะถึงแล้ว แล้วจะทำยังไงดีหละ ?

ทางแก้ทางหนึ่งที่ปัจจุบันทั่วโลกมุ่งพัฒนากันอยู่ก็คือ การเอาโปรเซสเซอร์ย่อย ๆ เหล่านี้มาช่วยกันคิด ช่วยกันทำ คือทำเป็นแบบทีมเวอร์ค ซึ่งแค่พูดก็เป็นเรื่องง่าย ๆ แต่การทำจริง ๆ นั้นยากมากทีเดียว เพราะจะต้องประสานกันทำงานให้ได้โดยต้องไม่เกิดการแย่งกันใช้ทรัพยากรของระบบเช่นหน่วยความจำหรือเพอริเฟอรอลต่าง ๆ เช่นสมมติมีโปรเซสเซอร์ 2 ตัวคือ P1 และ P2 ต้องการใช้หน่วยความจำร่วมกันตำแหน่งหนึ่งเพื่อทดเลข ถ้าต่างคนต่างเขียนข้อมูลของตนลงไป ผลก็คือข้อมูลของตัวที่เขียนลงไปก่อนจะหายไป เพราะถูกตัวหลังเขียนทับ ดังนั้นตัวที่เขียนก่อนเมื่ออ่านขึ้นมาจะได้ข้อมูลที่ไม่ใช่ตัวเองเขียนลงไป เป็นผลให้การคำนวณผิดพลาด ดังนั้นจึงต้องมีกลไกบางอย่างในการแก้ปัญหาลักษณะอย่างนี้ และนี้เองที่ทำให้ระบบที่มี n โปรเซสเซอร์จะมี performance ได้ไม่ถึง n เท่าของเครื่องที่มี 1 โปรเซสเซอร์

## ประเภทของ Computer

เอาละ หลังจากที่เกริ่นถึงเรื่องน่าปวดหัวของการใช้โปรเซสเซอร์หลายตัวมาช่วยกันทำงาน ก็มาถึงประเภทของคอมพิวเตอร์บ้าง โดยใช้วิธีแบ่งตามจำนวนของคำสั่ง (instruction stream) และข้อมูล (data stream) ที่ทำงานได้พร้อม ๆ กันในขณะใดขณะหนึ่ง หลายคนยังสับสนถึงคำว่า "multiprocessing" กับ "parallel processing" บางคนบอกว่ามันเหมือนกัน ในขณะที่บางคนบอกว่ามันต่างกัน จะว่ากันไปแล้วคำว่า

"multiprocessing" หมายถึงการทำงานของ processor หรือซีพียูจำนวนหนึ่งในระบบ เพื่อให้เกิดผลงานรวมที่ดีที่สุดสำหรับระบบ ส่วน "parallel processing" คือการทำงานพร้อม ๆ กันเหมือน ๆ กันของ processor หรือซีพียูหลาย ๆ ตัวสำหรับงาน ๆ หนึ่งแบบขนานกันไป ฟังดูก็ยังอาจสับสนอยู่ เรื่องการแบ่งประเภทของคอมพิวเตอร์ในลักษณะอย่างนี้ ก็มีหลายคนแบ่ง ๆ กัน แต่ที่เป็นจะยอมรับกันอย่างกว้างขวางก็คือของ Michael J. Flynn เขาแบ่งประเภทมาตั้งแต่ประมาณปี 1966 โดยแบ่งตามจำนวนสายของ instruction และ data ออกได้เป็น 4 ประเภทตามตารางในรูปที่ 1

1. SISD (Single instruction stream, single data stream) เป็นคอมพิวเตอร์ที่ทำงานโดยใช้คำสั่งสายเดียวกับข้อมูลสายเดียว เครื่องจะทำงานทีละคำสั่งกับข้อมูลชุดหนึ่ง ก็เป็นคอมพิวเตอร์ที่เรารู้จักคุ้นเคยกัน เช่น IBM 370, VAX หรือ Macintosh เป็นต้น

2. SIMD (Single instruction stream, multiple data stream) ตัวอย่างชัด ๆ คือพวก array processor คือพวกที่ใช้คำสั่งชุดเดียวกันกับข้อมูลมากกว่า 1 ชุดในขณะใดขณะหนึ่ง

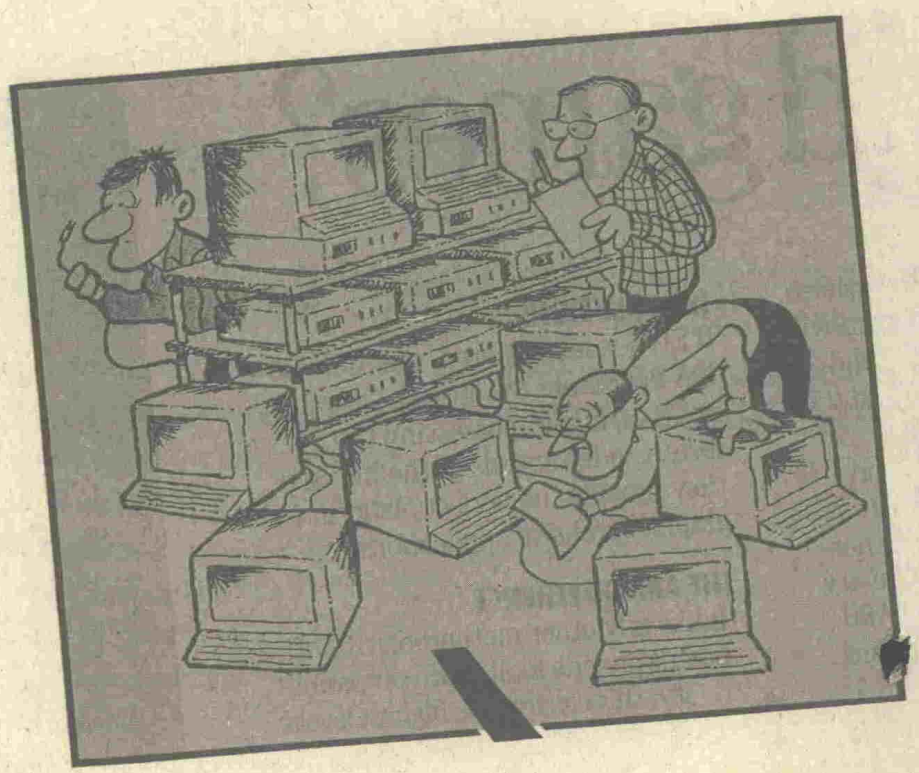
3. SIMD (Multiple instruction stream, single data stream) คือคอมพิวเตอร์ที่ทำงานโดยใช้คำสั่งหลาย ๆ ชุดพร้อม ๆ กันโดยใช้ทำงานข้อมูลชุดเดียว ฟังดูก็แปลก ใช่แล้ว ที่จริง ยังไม่ใคร่ใช้งานคอมพิวเตอร์ประเภทนี้กันจริง ๆ

4. MIMD (Multiple instruction stream, multiple data stream) คือคอมพิวเตอร์ที่ทำงานได้หลาย ๆ คำสั่งโดยใช้ข้อมูลได้หลายชุดพร้อม ๆ กัน แบบนี้แหละที่ชัดเจนที่สุดว่าเป็น multiprocessor

| ชื่อย่อ | ความหมาย | Instruction Stream | Data Stream | ตัวอย่าง |
|---|---|---|---|---|
| SISD | Single instruction stream , single data stream | 1 | 1 | IBM370, DEC, VAX |
| SIMD | Single instruction stream , multiple data stream | 1 | >1 | ILLIAC IV, NASA's MPP |
| MISD | Multiple instrct. stream , single data stream | >1 | 1 | Not used |
| MIMD | Multiple instrct. stream , multiple data stream | >1 | >1 | Cray X/MP, Butterfly |

รูปที่ 1 ตารางแสดงประเภทของคอมพิวเตอร์ แบ่งโดย Michael J. Flynn

ต่อมาเมื่อคอมพิวเตอร์มีมากแบบมากเครื่องขึ้น ก็พบว่าเกิดปัญหาในการจัดกลุ่มตามแบบของ Flynn ในปี 1988 David B. Skillicorn จาก Queen's University at Kingston รัฐ Ontario ได้ประกาศประเภทของคอมพิวเตอร์โดยแบ่งตามจำนวนและความสัมพันธ์ของคำสั่ง, ข้อมูลและรูปแบบของหน่วยความจำได้เป็น 28 แบบ และในปี 1990 Ralph Duncan จาก Control Data ได้เสนอแนะประเภทของ parallel computer ใหม่ โดยได้ใช้รากฐานจากของ Flynn และได้เพิ่มเติมประเภทใหม่ ๆ ลงไปให้สามารถครอบคลุมคอมพิวเตอร์ที่เกิดขึ้นภายหลังซึ่งไม่สามารถจัดเข้ากลุ่มของ Flynn ได้อย่างเหมาะสม



จากรูปที่ 2 parallel computer ถูกแบ่งออกเป็น 3 กลุ่มใหญ่ ๆ คือ Synchronous, MIMD และ MIMD-based paradigms

### Synchronous

ชื่อก็บอกชัดแล้ว คือทำงานแบบ parallel โดยมีการควบคุมจังหวะซึ่งกันและกัน (synchronously) ตัวอย่างเช่น

Pipelined Vector Processor คำว่า pipeline ก็เหมือนท่อ คือแต่ละคำสั่งของภาษาเครื่องจะถูกส่งเข้าคอมพิวเตอร์ทำงานโดยแบ่งเป็นขั้น ๆ เมื่อครบขั้นจึงจะเสร็จ 1 คำสั่ง ดังนั้นสมมติคำสั่งแรก $I_1$ ถูกส่งเข้าไปทำงานถึงขั้นที่ 2 คำสั่งถัดไปคือ $I_2$ จะถูกส่งเข้าไปในคอมพิวเตอร์ทำงานพร้อม ๆ กันทั้ง $I_1$ และ $I_2$ ที่คนละขั้น (step) กัน ลักษณะคล้าย ๆ กับการที่สินค้าถูกส่งเข้าในสายการผลิต (assembly line) ในโรงงาน ส่วน vector หมายถึงจะมี vector instruction ที่บ่งบอกถึงวิธีทำงาน (operation) และกำหนดกลุ่มของข้อมูลที่ต้องการที่เรียกว่าเวคเตอร์ เมื่อ vector instruction ถูก execute เวคเตอร์จะถูกส่งเข้าไปใน pipeline ที่เหมาะสมทีละตัว และทำงานพร้อม ๆ กัน ตัวอย่างเช่น เครื่อง Cray-1 มี 13 pipelines ทำงานขนานกันไปซึ่งถูกกำหนดไว้ให้ทำการคำนวณสำหรับเลข floating point เป็นต้น

Processor array เป็นพวก SIMD คือ ทำงานกับข้อมูลหลายชุดพร้อม ๆ กัน โดยใช้คำสั่งเดียวกัน ตัวอย่างงานที่เห็นได้ชัด ๆ ก็เช่น electronic spreadsheet นั่นเอง สำหรับตัวอย่างเครื่องก็เช่น MPP (Massive Parallel Processor) ที่ Goodyear Aerospace ออกแบบให้ NASA สำหรับงาน image processing ทำงานด้วยโปรเซสเซอร์ถึง 16,384 ตัว

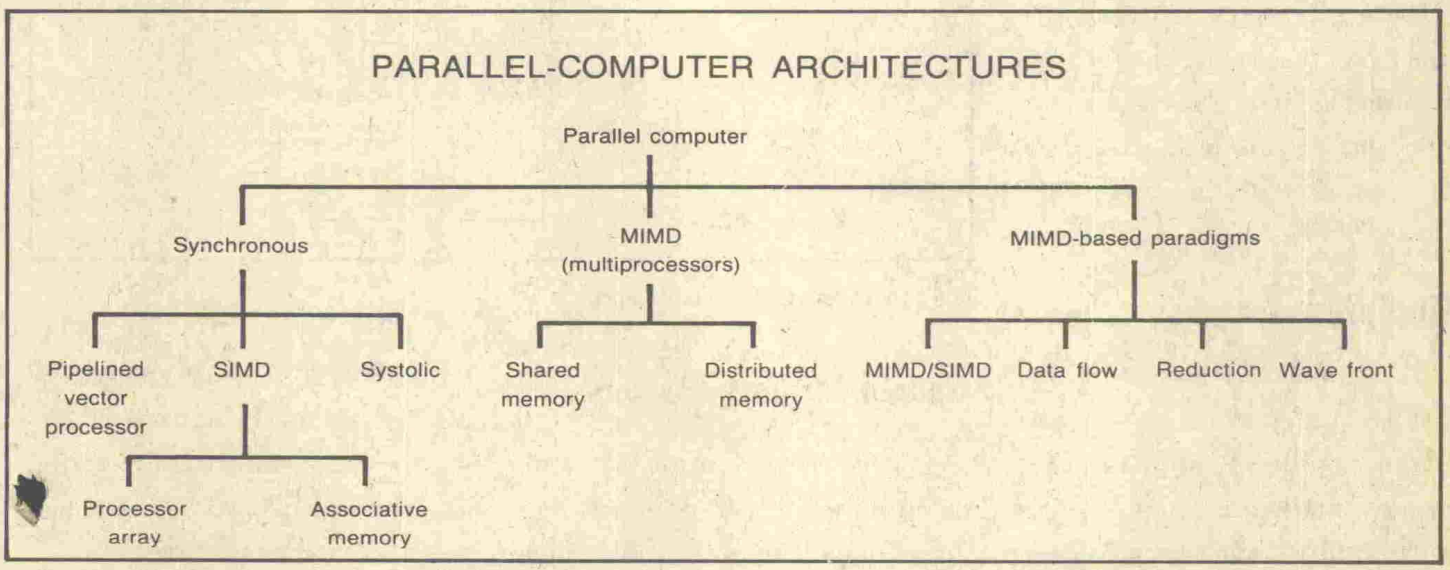
### MIMD

MIMD หรือ multiprocessor เป็นคอมพิวเตอร์ที่มีโปรเซสเซอร์หลายตัวทำงานกับข้อมูลหลาย ๆ ชุด โดยอิสระกัน แต่ก็ไม่ใช่ว่าต่างคนต่างทำ โปรเซสเซอร์ทุกตัวจะต้องมีการติดต่อระหว่างกัน ใช้ข้อมูลบางส่วนร่วมกัน และ software จะเป็นผู้ช่วยประสานงานให้ ถ้าโปรเซสเซอร์ตัวใดเสียหรือหยุดทำงาน ตัวที่เหลือจะรู้ ระบบดำเนินการจะหยุดการใช้งานโปรเซสเซอร์ตัวนั้น และจะแบ่งงานที่มีให้กับตัวที่เหลือ

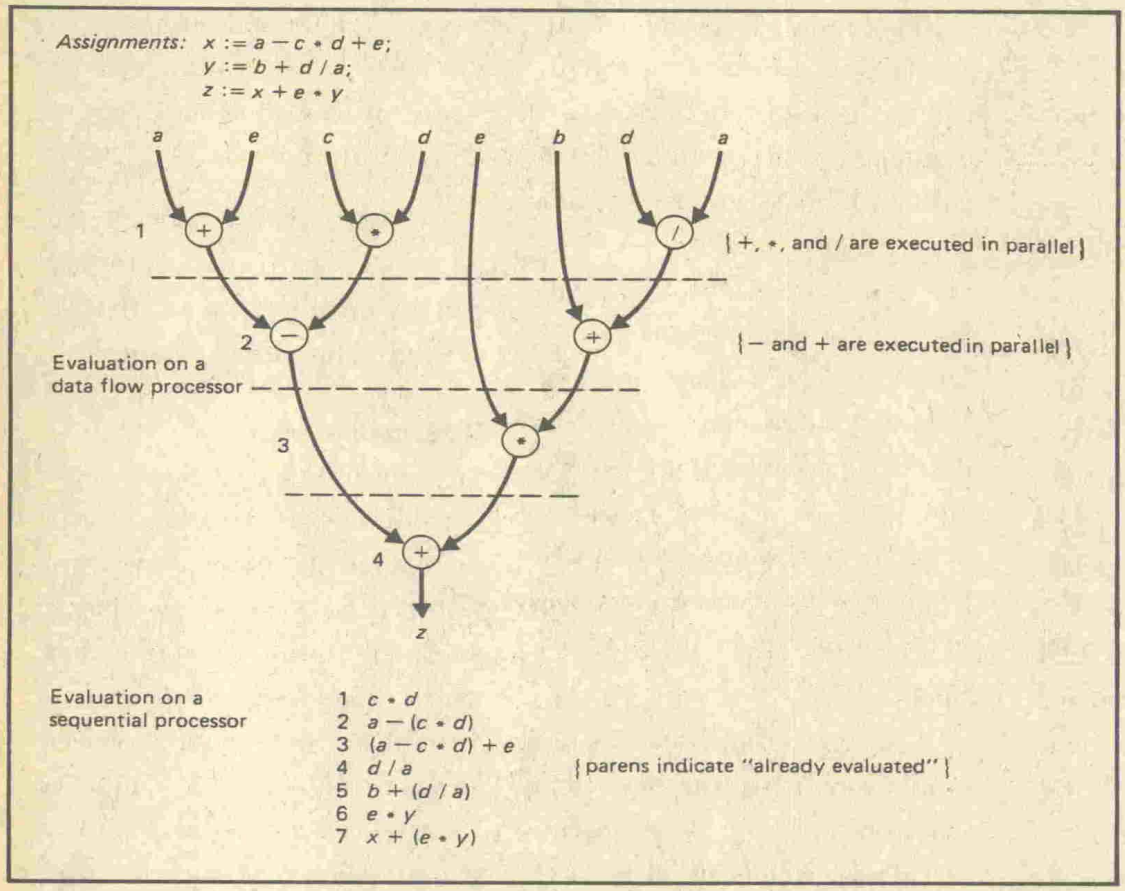
MIMD แบ่งย่อยออกเป็น 2 กลุ่ม คือ Share-memory กับ Distributed สำหรับรายละเอียดของเรื่อง MIMD นี้จะขอเก็บไว้ในตอนท้ายของบทความนี้

### MIMD-based paradigms

แบบนี้คล้าย ๆ กับ MIMD แต่จะเพิ่มหลักการจัดการเฉพาะบางอย่างเข้าไป ตัวอย่างเช่นคอมพิวเตอร์ประเภทที่เรียกว่า Data Flow Computer ข้อมูลจะเป็นตัวควบคุมการทำงานคือ การกระทำ (operation) จะเกิดขึ้นเมื่อมีข้อมูลพร้อม และอาจเกิด operation ได้พร้อม ๆ กันมากกว่า 1 ก็ได้ Data Flow Computer ยังไม่มีใช้กันในทางธุรกิจแต่หลักการของมันมีประโยชน์สำหรับการพัฒนา compiler ที่จะสร้างโค้ดที่ดีสำหรับเครื่องที่เป็น parallel ทั้งหลาย

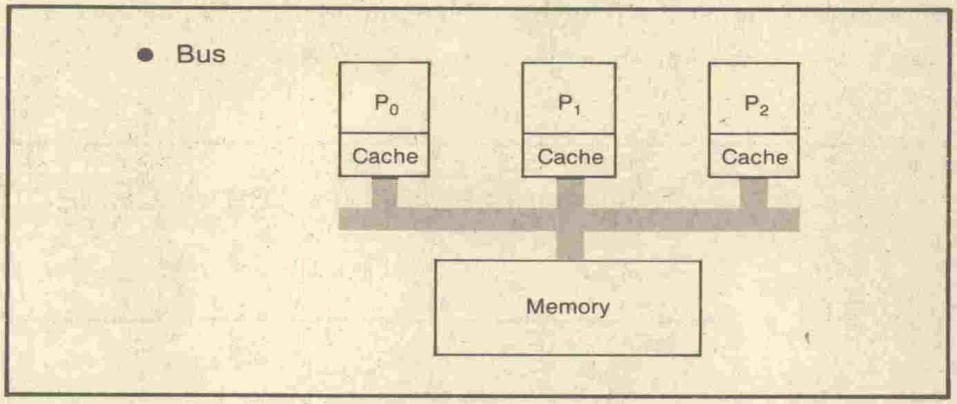



รูปที่ 2 การแบ่งประเภทคอมพิวเตอร์โดยนาย Ralph Duncan




รูปที่ 3 แสดงการทำงานแบบ Data flow ในส่วนบนเทียบกับการทำงานแบบ SISD ในส่วนล่าง

จากรูปที่ 3 สมมติจะต้องคำนวณหาค่า x, y, และ z ถ้าเป็นเครื่องชนิด SISD จะต้องทำถึง 7 ขั้น อย่างที่แสดงไว้ในตอนล่างของรูป แต่ถ้าเป็นเครื่อง Data Flow Computer จะยุบการทำงานเหลือเพียง 4 ขั้น โดยขั้นที่ 1 จะทำการ +, *, และ / พร้อมกัน และขั้นที่ 2 จะทำ − และ + พร้อมกัน




รูปที่ 4 Shared-memory systems แบบ Shared bus

## Multiprocessor

อย่างที่เกริ่นมาในหัวข้อของ MIMD ว่า MIMD หรือ multiprocessor แบ่งย่อยออกเป็น 2 กลุ่มคือ Share-memory กับ Distributed ตอนนี้ก็จะมาแจกแจงประเภทของ multiprocessor กันให้ลึกลงไปอีกนิดหน่อย

## Shared-Memory Systems

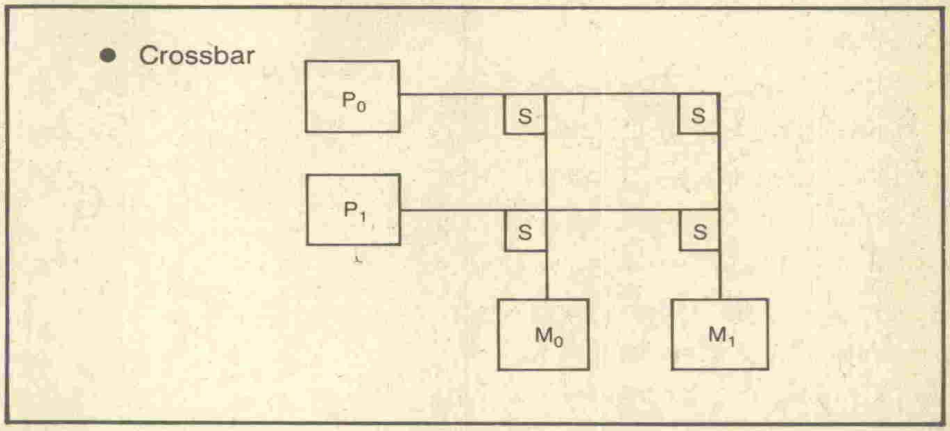
โปรเซสเซอร์ทั้งหลายในระบบนี้จะติดต่อกันโดยการอ่าน-เขียนหน่วยความจำที่เป็นของส่วนรวมหรือกองกลางในลักษณะของ shared bus หรือ crossbar หรือแบบ multi stage interconnection network ดังแสดงไว้ในรูปที่ 4-6

1. Shared Bus ในรูปที่ 4 จะเห็นว่า มีการใช้เส้นทางการติดต่อเป็น bus ชุดหนึ่งระหว่างทุกโปรเซสเซอร์, หน่วยความจำและหน่วยควบคุม I/O (I/O

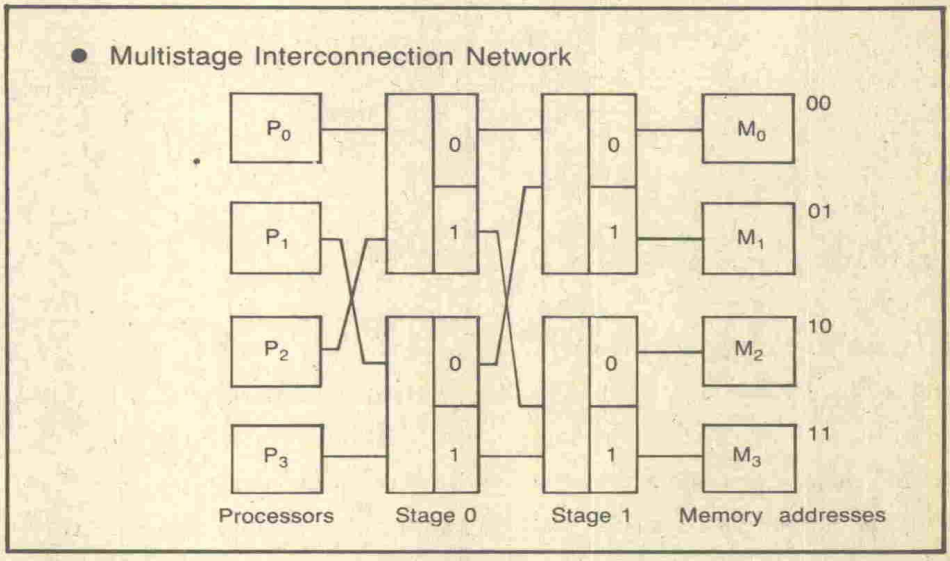
processor) ซึ่งจะเรียกแต่ละตัวว่า unit การใช้ bus นี้ก็คล้าย ๆ กับเน็ตเวอร์กแบบ Ethernet ซึ่งใช้ shared bus เช่นกัน ในที่นี้ โปรเซสเซอร์หรือ I/O processor ตัวไหนต้องการจะถ่ายเทข้อมูล จะต้องตรวจสอบก่อนว่า มีใครใช้ bus อยู่ก่อนหรือเปล่า เพื่อไม่ให้เกิดการ jam กันของข้อมูล ส่วนฝ่ายรับก็จะต้องสามารถเลือกได้ว่าข้อมูลที่อยู่บน bus นั้น ผู้ส่งส่งให้เราหรือเปล่า ถ้าใช่ จะต้องอ่านและตีความสัญญาณข้อมูลเหล่านั้น ข้อดีของแบบ shared bus คือสามารถเพิ่มเติม unit ใหม่เข้าไปได้ง่าย เพียงเพิ่มการติดต่อระหว่าง bus กับ unit ใหม่ ข้อเสียคือการมีเส้นทางติดต่อระหว่าง unit มีเพียง bus เดียวทำให้เกิดการติดต่อได้ไม่เกินหนึ่งคู่ในเวลาใดเวลาหนึ่ง และถ้า bus นั้นเกิดเสียหายขึ้นมา ทั้งระบบจะหยุดทันที นอกจากนั้นยังมีข้อจำกัดของความเร็วอยู่ที่ความเร็วของ bus และเรื่องของความแออัดในการใช้ bus เมื่อจำนวนโปรเซสเซอร์เพิ่มขึ้น

ข้อจำกัดเหล่านี้ทำให้มีการใช้ shared bus สำหรับโปรเซสเซอร์จำนวนไม่มากนัก และบางทีก็แก้โดยเพิ่มจำนวน bus ขึ้น

2. Crossbar เมื่อเพิ่มจำนวน shared bus ให้กับระบบมากขึ้น ก็จะเกิด multiprocessor ประเภท crossbar หรือชื่อเต็ม ๆ ว่า crossbar-switch matrix ขึ้น ซึ่งจะมีเส้นทางเฉพาะสำหรับทุก ๆ unit ประสานกันเป็นตาราง ดังในรูปที่ 5 การติดต่อระหว่าง unit จึงเกิดได้มากกว่า 1 คู่พร้อม ๆ กันโดยใช้เส้นทางไม่ซ้อนกัน หรืออาจส่งข้อมูลได้พร้อมกันหมดทุก storage unit เลยก็ได้ ที่รอยตัดของ bus หรือเส้นทางจะมี switch ซึ่งทำหน้าที่ป้องกันการแย่งกันหรือตีกัน (contention)




รูปที่ 5 Shared-memory systems แบบ Crossbar




รูปที่ 6 Shared-memory systems แบบ Multistage Interconnection Network

เมื่อเกิดการอ้างถึงข้อมูลใน storage unit เดียวกันพร้อมกันโดย function unit มากกว่า 1 ตัว

3. Multistage Interconnection Network (MIN) ชื่อก็บอกชัดแล้วว่าเป็นการต่อกันแบบหลายขั้น (multistage) โปรเซสเซอร์ทั้งหลายจะติดต่อกับ storage unit หรือส่วนของหน่วยความจำทุกยูนิตโดยผ่านสวิตช์หลายขั้น ในรูปที่ 6 โปรเซสเซอร์ P1 จะใช้งานหรืออ้างอิงถึงหน่วยความจำในยูนิต M3 ด้วย address 11 สวิตช์อันแรกจะใช้บิตสูงหรือ most significant bit (MSB) ของ address ส่วนสวิตช์ตัวหลังจะใช้บิตต่ำหรือ least significant bit (LSB) สังเกตได้ว่า การติดต่อระหว่างโปรเซสเซอร์ N ตัว โดย N เท่ากับ 2 ยกกำลังด้วยเลขบวกที่มากกว่า 1 จะต้องมี $\log_2 N$ ขั้น และแต่ละขั้นจะประกอบด้วยสวิตช์ N/2 ตัว

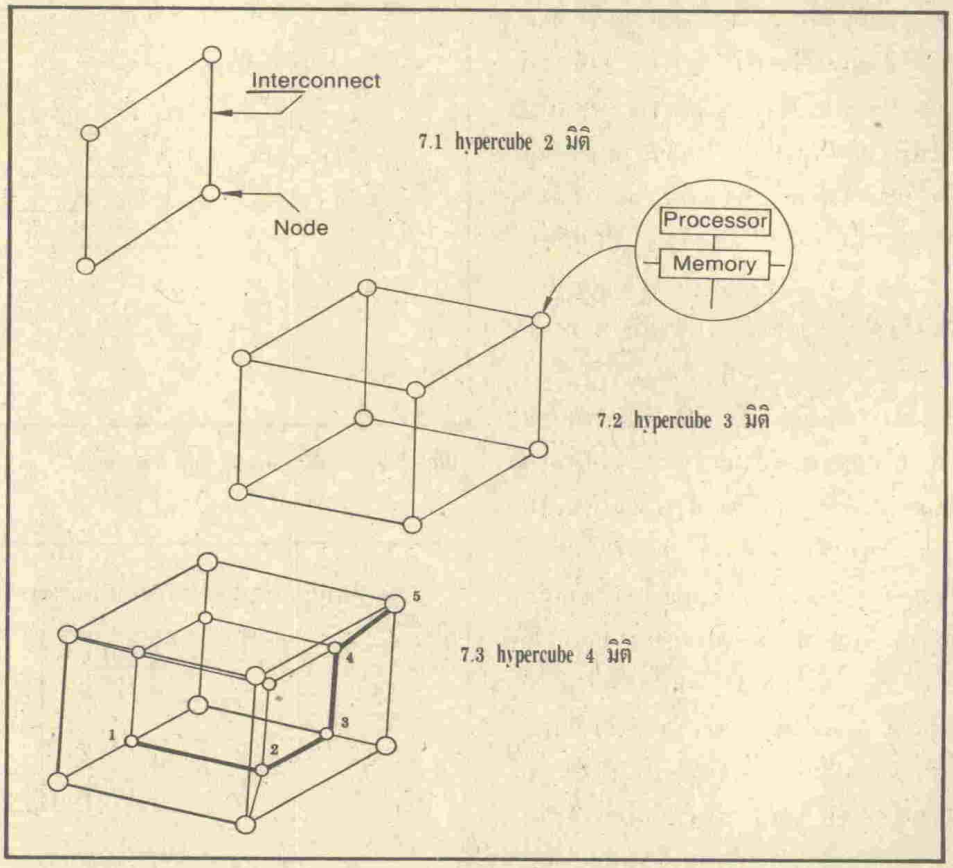
## Distributed Memory Systems

มัลติโปรเซสเซอร์แบบนี้ โปรเซสเซอร์แต่ละตัวไม่ได้ใช้หน่วยความจำ

ร่วมกันเหมือนในแบบ shared-memory จึงลดปัญหาเรื่องการแย่งกันหรือตีกัน และรวมทั้ง consistency ของข้อมูล ทำให้มัลติโปรเซสเซอร์แบบนี้มีข้อดีกว่าแบบ shared-memory มากและมี scalability ที่ดีกว่าด้วย scalability หมายถึงเมื่อเพิ่มจำนวนโปรเซสเซอร์เข้าไปในระบบ จะมีผลไปเพิ่ม performance ของระบบได้แค่ไหน อย่างดีที่สุดคือเป็นแบบเส้นตรง คือถ้าเพิ่มจำนวนโปรเซสเซอร์เข้าไป 10 เปอร์เซ็นต์ ก็จะได้ performance เพิ่มขึ้นอีก 10 เปอร์เซ็นต์ ในระบบ shared-memory จะมี scalability เฉพาะเมื่อเพิ่มจำนวนโปรเซสเซอร์จำนวนน้อย หากเพิ่มเกิน 20-30 ตัว เราจะเห็นอัตราส่วนของ performance ต่อโปรเซสเซอร์ตกลง เนื่องจากความแออัดในการใช้ bus

ส่วนใน distributed จะมีหน่วยความจำส่วนตัวติดต่อกับโปรเซสเซอร์ตัวอื่น ๆ ผ่านระบบข่าวสารให้เกิดความสอดคล้องกันในการทำงาน ดังนั้นอาจจะเรียกคอมพิวเตอร์ประเภทนี้ว่า "message-passing multiprocessors" และเพราะแต่ละโปรเซสเซอร์จะมีหน่วยความจำและกระบวนการติดต่อกับโปรเซสเซอร์ตัวอื่นเป็นของมันเอง ซึ่งเรียก 3 หน่วยย่อยนี้รวมกันว่า node แต่ละโหนดจะมีความสมบูรณ์ในตัวเองคล้าย ๆ กับเป็นคอมพิวเตอร์ตัวหนึ่ง ดังนั้นระบบนี้จึงมีอีกชื่อหนึ่งว่า "multicomputer"

การเชื่อมต่อระหว่างแต่ละโหนดในระบบนี้มีหลายแบบ เช่น แบบวงแหวน (ring), ต้นไม้ (binary-tree), และอะเรย์ 2 มิติ (two-dimensional arrays of processor) ซึ่งโหนดแต่ละตัวจะเชื่อมต่อกับโหนดทุกตัวที่อยู่ข้าง ๆ มัน ปัญหาก็อยู่ที่ว่าการติดต่อระหว่างโหนดที่อยู่ไกลกัน จะต้องผ่านโหนดจำนวนมากขึ้นอย่างมาก เมื่อเพิ่มโหนดเข้าไปในระบบ



7.1 hypercube 2 มิติ

7.2 hypercube 3 มิติ

7.3 hypercube 4 มิติ

การแก้ปัญหาที่ดีที่สุดในปัจจุบันคือ การเชื่อมต่อแบบ hypercube โปรเซสเซอร์ N ตัวจะถูกจัดเป็นลูกบาศก์ n มิติโดย $N=2^n$ แต่ละโปรเซสเซอร์จะเชื่อมต่อกับตัวอื่น n ตัวและโปรเซสเซอร์ 2 ตัวที่อยู่ไกลกันที่สุดจะห่างกัน n โหนด

เริ่มต้นจาก hypercube 2 มิติ โหนด 4 กลุ่มต่อกันเป็นดังรูปที่ 7.1 ที่แต่ละมุมเป็นโหนดซึ่งประกอบด้วยโปรเซสเซอร์, หน่วยความจำ, และทางติดต่อกับโหนดอื่น โหนดแต่ละตัวจะต่อกับโหนดอื่น 2 กลุ่ม และโหนดที่อยู่ไกลกันที่สุดก็จะห่างกัน 2 ช่วงโหนด เช่น คู่ที่อยู่คนละมุมกันนั่นเอง

ในรูปที่ 7.2 เป็นรูปของ hypercube 3 มิติ ประกอบด้วยโหนดจำนวน $2^3=8$ กลุ่มเชื่อมต่อกันโดยแต่ละโหนดจะต่อกับโหนดอื่น 3 กลุ่ม และ 2 โหนดที่ไกลกันที่สุดจะห่างกัน 3 ช่วงโหนด ได้แก่คู่ของโหนดที่อยู่มุมตรงกันข้ามนั่นเอง

ในรูปที่ 7.3 เป็น hypercube 4 มิติ ก็คือ hypercube 3 มิติมาต่อกัน จะเห็นได้ว่า 2 โหนดที่ไกลกันที่สุดจะห่างกัน 4 ช่วงโหนด เช่น ในเส้นทางที่แสดงเป็นสีดำ จาก node 1-2-3-4-5 ในรูป

ข้อดีของ hypercube จะเห็นได้ชัด ๆ เมื่อต่อโปรเซสเซอร์จำนวนเป็น 100 หรือเป็น 1000 ตัว เช่นขนาด 8 มิติ จะประกอบด้วยโปรเซสเซอร์ $2^8=256$ ตัว การติดต่อระหว่างโปรเซสเซอร์คู่ที่ไกลกันที่สุดจะผ่านโหนดเพียง 8 โหนด หรือถ้าเป็นขนาด 1024 ตัวหรือ 10 มิติ ก็จะไกลกันเพียง 10 โหนดเท่านั้น